

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problems Mailbox.**

THIS PAGE BLANK (USPTO)



Canal+ David Hawkes – Le Ponant C - 23, rue Leblanc – 75015 PARIS
Tél. : 01 53 78 62 13 – Fax. : 01 44 25 78 72

85/89 QUAI ANDRE CITROEN 75711 PARIS CEDEX 15
TEL 01 44 25 10 00 FAX 01 44 25 12 34

MATHYS & SQUIRE
Mr. Andy BOOTH
100 Gray's Inn Road
LONDON WC1X 8AL
UNITED KINGDOM

D.H.L.

Paris, September 3, 1998

Dear Andy,

Thanks for the draft specification. I have now had a chance to go through this with Denis Choulette and his annotated version of the specification is enclosed. As you will see, he has added a great deal of useful information.

My comments are the following:

Page 3, paragraph 3 to last paragraph:

Organising the files over the boundaries of RAM, ROM and FLASH is something that may be known in the prior art but is far from being typical and I would hesitate to describe Figure 3 as "prior art".

It would seem more logical to discuss the genealogical relationship in a normal file arborescence with reference to Figure 2 and in relation to file organisation in a single memory.

Page 4, paragraph 1:

The word "identifier" is used for files both the standard file organisation and the hierarchy according to the invention (see page 4, last paragraph). Given the generic nature of this term, this is not incorrect, but it is imprecise.

In the prior art, the identifier in a "father" file is an address or pointer to an address of the "son" file. In the invention, the identifier is a unique number in a son file associated with the father. This enables the address memory location of, for example, a "father" file to be changed whilst retaining sufficient information to enable the arborescence to be re-created from information in a "son" file.

Page 4, paragraph 3:

As far as I know, a grandfather directory usually only contains the address of its sons, and does not contain a "grandson" identifier. See also paragraph 4, last line.

Page 4, paragraph 5:

See Denis's comments regarding the advantages of the invention in relation to security and consistency of data. The invention is particularly useful in memory volumes where writing data is time consuming; i.e. FLASH memories.

In view of the need to run through the entire memory to "reverse engineer" the actual arborescence, the invention is not particularly useful for other types of memory (e.g. RAM or EEPROM) and, in the system actually used, file data in other memories is organised conventionally (father addressing son). I did not know this!

Pages 4 and 5 (Claim 1):

See my comments regarding "identifier". To be clarified in a sub-claim ?

Page 5, lines 13 to 16:

It is not clear that this corresponds to what is actually done. If the aim is to describe the run through of the data in the FLASH to create a virtual arborescence in the RAM, this needs to be clarified.

A sub-claim could also be introduced specifying that the file and directory data organised according to the son/father relation of the invention is exclusive to the FLASH memory of the system.

Page 5, paragraph 6:

I would prefer to avoid cut and stick repetition of method and apparatus claims in the introduction. I note that you have already included, for example, a catch-all statement on page 13 regarding the interchangeability of method and apparatus claims.

Page 6, last paragraph to page 7, paragraph 1:

Variable block size may indeed be known in some systems, but this is far from being how FLASH memories are generally organised. If anything, the inverse is true, most FLASH memories being organised with a fixed size of memory blocks. As a corollary, compaction of memory blocks when a page is updated is not usually carried out in conventional systems since block size is constant and there is no advantage from re-arranging the order of the blocks. A valid block thus usually remains fixed in the same position in the page when the page is updated.

Page 7, lines 11 to 15:

I do not think this adds much to the description and I would prefer to just describe the prior art system that is used in Mediahighway (lines 15 to 18).

Page 8, lines 5 to 14:

I am not sure of the need to specify that each page of the memory contains a plurality of blocks; one can easily imagine a memory in which some of the pages are occupied by a single memory block. Accordingly, it might be simpler to describe a memory including a source page containing valid and invalid blocks and a designated transfer page.

I am also not sure if it is quite correct to refer to the transfer page as comprising "freeblocks". My understanding is rather that this page is virgin. See Figure 12.

Finally, there is something about "at one end" that bothers me! I would prefer to refer to a compacting of the blocks being carried out during the copying operation. After all, one could (just about) imagine sending half the blocks to one end and half to the other end, since the space in between would still be the same as if they are all at one end.

Page 8, lines 20 to 30. Not possible to redraft as a dependant claim?

Page 9, paragraph 4:

The use of variable block size is practically essential for the compaction operation to have any utility. To be expanded?

I would also mention as a sub-claim that the source page is deleted to be in order to become the new buffer page immediately after the transfer is carried out; i.e. long before the next transfer. This enables a reduction in processing time. See my letter of 18 June.

I would also specify in a sub-claim that a valid block is changed to an invalid block by changing (e.g. from 1 to 0) the value of a one-bit flag. This takes advantage of the fact that, in a flash memory, a deletion of a bit (1 to 0) can be carried out without requiring re-writing of page, unlike the writing of a bit (0 to 1). See my letter of 18 June. This should also be included in the description, if not already present.

Similarly, as an intermediate position between "memory" and "a flash memory", a sub-claim to the applicability of the invention to a memory of a kind that is not freely writable might also be useful. The same comments apply to the other claims.

Page 10, paragraph 3:

I have no problems with this as a broad claim 1, but my feeling is that this will read onto any conditional access system. In addition to the description of the access rights as being stored in a header for the data as a sub-claim, I would also suggest sub-claims referring to the data as being stored in a series of files, the files being organised in a hierarchy etc.

Page 10, lines 23 and 24:

"One or more identifiers for the parties..."

Page 10, last paragraph:

Not possible to redraft as a dependant claim?

Description:

See in particular Denis's comments.

Page 18, paragraph 2:

Figure previously identified as prior art.

Page 20, line 3:

Father ID=0.

Page 20, paragraph 4:

I would imagine that the RAM chart is updated at any change of the data structure in the FLASH memory.

Page 20, paragraph 5:

As noted above, in the actual embodiment, only the FLASH memory is organised as son/father, the other memories (ROM, RAM) using the conventional father/son organisation.

Page 22, last paragraph:

Should read "Figure 8".

Page 23, lines 20, 21:

See Denis' explanation as to why a NEW state is always necessary.

Page 24, line 21 to page 25, line 17:

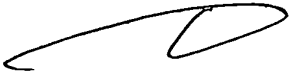
See Denis' comments.

Page 25, last paragraph:

As I understand it, the cumulative invalid memory size will always be the determining factor in deciding which page to transfer. I would suggest deleting Figure 11 completely and replacing it with Figure 13.

If you need any further clarifications, I suggest you call Denis Choulette (33.1.44.25.79.85). Generally, however, the application seems in pretty good shape.

Best regards,

A handwritten signature in black ink, appearing to be 'D. HAWKES', with a long horizontal stroke extending to the left.

Dave HAWKES

Encl.

MANAGEMENT OF DATA IN A RECEIVER/DECODER

5 The present invention relates to the management of data. In particular, the invention relates to the following aspects of managing data, for example, in a receiver/decoder:

- recording the arborescence of data stored as files and directories in a memory;
- transferring blocks of data between pages of memory to enable fresh data to be stored in the memory; and
- restricting data access in a receiver/decoder.

10

The term "receiver/decoder" used herein may connote a receiver for receiving either encoded or non-encoded signals, for example, television and/or radio signals, which may be broadcast or transmitted by some other means. The term may also connote a decoder for decoding received signals. Embodiments of such receiver/decoders may include a decoder integral with the receiver for decoding the received signals, for example, in a "set-top box", such a decoder functioning in combination with a physically separate receiver, or such a decoder including additional functions, such as a web browser, a video recorder, or a television.

20 In a broadcast digital television system, received signals are passed to a receiver/decoder and thence to a television set. As used herein, the term "digital television system" includes for example any satellite, terrestrial, cable and other system. The receiver/decoder decodes a compressed MPEG-type signal into a television signal for the television set. It is controlled by a remote controller handset, through an interface in the receiver/decoder. The receiver/decoder is used to process the incoming bit stream, and includes a variety of application modules which cause the receiver/decoder to perform a variety of control and other functions.

30 The term MPEG refers to the data transmission standards developed by the International Standards Organisation working group "Motion Pictures Expert Group" and in particular but not exclusively the MPEG-2 standard developed for digital television applications and set out in the documents ISO 13818-1, ISO 13818-2, ISO

13818-3 and ISO 13818-4. In the context of the present patent application, the term includes all variants, modifications or developments of MPEG formats applicable to the field of digital data transmission.

5 Such a receiver/decoder as described above may have a variety of devices coupled to it, such as a card reader for the user to pass an authorization card through to confirm which services the user is authorized to use, a hand-held television receiver control wand, a television display unit, and a second card reader for use with bank cards to allow the user to perform home banking functions. It may also have a variety of ports
10 coupled to it, for example, a modem for access to the Internet and for conducting home banking transactions.

With reference to Figure 1, the receiver/decoder includes a central processor 50 and associated memory volumes 54, 56 and 58. The memory volumes may be coupled
15 directly to the central processor 50, or, as shown in Figure 1, may be coupled to the central processor over a bus 52.

A variety of types of memory volumes are available. One major distinction between different kinds of memory volume is between volatile and non-volatile memory.
20 Volatile memory retains its contents only while power is supplied to the memory, losing its contents as soon as its power supply is disconnected, while non-volatile memory retains its contents indefinitely even if its power supply is disconnected. The other major distinction is between writable memory and read-only memory.

25 Volatile memory is generally known as RAM, while there are various kinds of non-volatile memory. RAM is normally writable, while read-only memory is known as ROM. This latter distinction is not necessarily hard-and-fast. Any memory must of course be writable in some sense at least once, but some kinds of ROM-like memory can have their contents changed, albeit with some difficulty. Thus there are
30 memory types such as EEPROM (electrically erasable programmable read-only memory), and Flash memory.

The different kinds of memory have different characteristics (for example, different read times and different costs), so it is often desirable to use a combination of several different kinds in a single receiver/decoder, as illustrated in Figure 1.

5 Figure 2 illustrates the organisation of data in a memory volume. The data is arranged in an arborescent arrangement of directories and files. As discussed in more detail below, each directory containing a list of identifiers for all of the files and/or directories which are identified by that directory for tracing the arborescence of the contents of the data.

10

Data may be divided across two or more of the memory volumes of the receiver/decoder. Figure 3 illustrates a typical organisation of directories D and files F of data across RAM volume 54, ROM volume 56 and Flash volume 58 of a receiver/decoder.

15

Directory D5 identifies files F5 and F6 stored in ROM volume 56. Thus, directory D5 is stored in ROM volume 56 only.

20

Directory D4 identifies file F3 stored in Flash volume 58, and file F4, directory D5 and files F5 and F6 (that is, the files identified by directory D5) stored in ROM volume 56. Thus, directory D4 is stored in both Flash volume 58 and ROM volume 56.

25

Directory D2 identifies file F2 stored in RAM volume 54 and directory D4 (and therefore all of the directories and files identified thereby) stored in the Flash volume 58 and ROM volume 56. Thus, directory D2 is stored in RAM volume 54, Flash volume 58 and ROM volume 56.

30

Directory D1 identifies file F1 stored in RAM volume 54, directory D3 stored in Flash volume 58, and directory D2 (and therefore all of the directories and files identified thereby) stored in RAM volume 54, Flash volume 58 and ROM volume 56. Thus, directory D1 is stored in RAM volume 54, Flash volume 58 and ROM volume 56.


The arborescence of the data organisation can be thought of in terms of the various generations of a genealogical family tree, being made up of "sons", "fathers", "grandfathers", and so on. Each directory contains an identifier for each of the members of the family which are its direct descendants.

5

Each father contains an identifier for each of its sons. For example, father directory D5 contains identifiers for files F5 and F6, which are its sons.

Each grandfather contains an identifier for each of its sons, and an identifier for each of its sons' sons. For example, grandfather directory D4 contains an identifier for each its sons, that is, identifiers for files F3, F4 and directory D5, and identifiers for each of its sons' sons, that is, identifiers for files F5 and F6.


10

Sure? 

15

The arborescent arrangement of directories and files is constantly being reorganised by the central processor 50, for example, if a file is moved between memory volumes. When a file is moved so that the arborescence of the data changes, then both the directories from which the file was previously directly descended and the directories from which the file is now directly descended must be modified for correct identification of the location of the file. For example, if file F3 is moved from Flash volume 58 to RAM volume 54 so that its new "father" is directory D1, then directories D1, ~~D2~~ and D4 must be modified. Depending on the number of ancestors which a file has, this can result in substantial modification of the directories of the data.

20

Not Sure 

25

The present invention seeks in one aspect to provide an improved system for tracing the arborescence of data in a memory volume which reduces the time required to update the arborescence when a file moves.

For use in memory device where writing is time consuming. For a RAM device Tree, the gain is not important.

30

In a first aspect, the present invention provides a method of recording the arborescence of data stored as files and directories in a memory, said method comprising the step of storing in association with each file and directory of the data an identifier of the directory, if any, immediately preceding that file or directory in the arborescence of

Also useful for consistency (see page 19)

the data.

5 This can provide for faster recording of the arborescence of the data, as each file or directory contains an identifier of the immediately preceding directory, and not identifiers of the directories and files which immediately follow the directory in the arborescence, as is conventional, as the number of directories or files which must be modified when the arborescence of the data can be significantly less than in the conventional system.

10 The identifier may be stored in a header of the file or directory. This provides for a convenient location of the file or directory in the arborescence of the data.

15 At least part of the data may be stored in a Flash memory volume. If so, the arborescence of that part of the data is preferably recorded in a RAM memory volume, and a header of a file is preferably stored in a dedicated block of Flash memory. The memory volumes may be provided in a receiver/decoder.

20 Preferably, the stored identifier is changed when a file or directory is moved to immediately precede another directory. This provides for quick modification of the arborescence of the data when a file is moved.

25 In a related aspect, the present invention provides apparatus for recording the arborescence of data stored as files and directories, said apparatus comprising means for storing in association with each file and directory of the data an identifier of the directory, if any, immediately preceding that file or directory in the arborescence of the data.

30 The storage means may be adapted to store the identifier in a header of the file or directory.

The apparatus may be provided in combination with a Flash memory volume and a RAM memory volume. If so, the apparatus may further comprise means for recording

in the RAM memory volume the arborescence of data stored in the Flash memory volume. The recording means may be conveniently provided by the central processor.

5 The apparatus may further comprise means for changing the stored identifier when a file or directory is moved to immediately precede another directory. The changing means may also be conveniently provided by the central processor.

This aspect of the invention also provides a receiver/decoder comprising apparatus as
aforementioned.

10

Flash memory is generally ROM-like, in that it is non-volatile. It is also intended to be used in a generally ROM-like manner, being read from but not written to. However, Flash memory can be written to, but only with some difficulty. Specifically, Flash memory is normally divided into pages, typically each of many
15 kilobytes in length, and writing to Flash memory is by page. In more detail, to re-use a block of Flash memory, an entire page has to be erased so that new data can be written in that block.

The information in Flash memory is organized into blocks of substantial size. A
20 block may comprise data, for example, tables of permanent or semi-permanent information, or a program or sub-routine. The block sizes will normally be chosen to be less than the page size (if a block is larger than a page, it will usually be feasible to split it into sub-blocks which are less than the page size).

25 Typically, when Flash memory is updated it is desirable to retain some of the information already in it. This therefore requires the page being updated to be read into RAM to form an image of the page; this image in RAM can then be updated by inserting whatever new information is to be entered into the page. At the same time, any information in the page which is no longer required can be deleted. The updated
30 image can then be written back into Flash memory.

In general, the block size will not be fixed; that is, different blocks will be different

sizes. This can cause difficulties when some existing blocks are to be discarded and fresh blocks are to be added. These difficulties can largely be overcome by allowing the blocks to be movable; when a page is updated, the blocks to be retained in the page are rearranged so that any unused areas on the page are merged into a single large unused area.

If the blocks are movable, then they cannot be addressed by fixed addresses. Instead, some sort of block locating or addressing data structure must be maintained so that the blocks can in effect be searched for by using some kind of name or descriptor.

With blocks of different sizes, this requires information about both the locations and the nature of the blocks. This is conventionally achieved in various ways. A detailed directory of the blocks (their locations and nature) can be maintained at the beginning of the Flash memory, or a location directory of the blocks can be maintained at the beginning of the Flash memory and each block can include a header giving the nature of the block. Alternatively, a separate block locating data structure can be maintained for each page of the Flash memory. A block locating data structure is held at least partially in an external memory outside the Flash memory itself, such as in an EEPROM memory.

It has been realized pursuant to the present invention that all these block locating data structures have the common characteristic that updating of the Flash memory is required for every change in its contents. To write a fresh block requires Flash memory updating; deletion of a block similarly requires updating. Although it is not necessary to physically delete the block, the block locating data structure has to be updated to indicate that the block is no longer valid.

Such a system for updating Flash memory has a number of drawbacks. First, an entire page of the Flash memory must be copied into RAM memory to enable fresh data to be stored in the Flash memory. Therefore, it is necessary to have a buffer in the RAM memory which is the same size as a page of Flash memory. Secondly, it is essential to have an EEPROM memory volume for storing the block locating data structure for identifying the location and status of blocks stored in the Flash memory.

The present invention seeks, in a second aspect, to provide an improved system for updating the contents of a Flash memory volume which does not require the use of ROM or RAM memory to effect updating.

5 In a second aspect, the present invention provides a method of transferring blocks of data between pages of memory to enable fresh data to be stored in said memory, said memory comprising a plurality of pages, each page of said memory comprising a plurality of blocks, said blocks comprising free blocks which contain no data, valid blocks containing valid data and invalid blocks containing invalid data, at least one
10 of the pages of memory being designated as a transfer page initially comprising only free blocks, the method comprising the steps of:

copying valid blocks from a designated page of memory into free blocks at one end of said transfer page; and

erasing the designated page.

15

By copying the valid blocks to one end, such as the start, of the transfer page, sufficient room for storing the fresh data can be left at the other end of the transfer page.

20 Thus, this aspect of the present invention also provides a method of storing data in memory, said memory comprising a plurality of pages, each page comprising a plurality of blocks, said blocks comprising free blocks which contain no data, valid blocks containing valid data and invalid blocks containing invalid data, at least one of the pages of memory being designated as a transfer page initially comprising only
25 free blocks, the method comprising the steps of:

copying valid blocks from a designated page of memory into free blocks at one end of said transfer page;

erasing the designated page; and

30 storing the data in at least one free block after the copied valid blocks in said transfer page.

Preferably, the erased designated page is redesignated as a new transfer page. This

can enable the aforementioned methods to be repeated when further data is to be stored in the memory.

5 In one preferred embodiment, a page is designated in dependence on the number of invalid blocks of the page. For example, the designated page may be the page having the most invalid blocks. This can provide for quick designation of the page from which valid pages are to be copied to the transfer page.

10 In another preferred embodiment, a page is designated in dependence on the cumulative size of the invalid blocks of the page. For example, the designated page may be a page having at least one invalid block having a cumulative size equal to or greater than the size of the fresh data. This can ensure that there will be sufficient room for storing the data in the transfer page once the copying of the valid blocks to the transfer page has been completed.

15

The blocks may have a variable size. This can enable data to be stored in a single block, in preference to storing the data within a plurality of blocks of fixed size.

20

Preferably, the memory comprises a Flash memory volume.

The memory may comprise a memory volume of a receiver/decoder, in which the data may be downloaded from a bitstream, preferably in the form of MPEG tables.

25

30

Access to data stored in the memory of the receiver/decoder may be required by a number of parties. One such party is the author of the data, who may require access to the data to correct an error in the data or to replace the data with an upgraded version. Another such party is the provider of an interactive application which uses the data. Whilst wanting such a provider to have access to the data so as to be able to read the data, thereby enabling the interactive application to use the data, the author of the data may want to prohibit the provider from over-writing the data with its own data. The provider may also want to prohibit any access to the data by, for example, the manufacturer or the owner of the receiver/decoder, thereby maintaining the

confidentiality of the data to these parties.

The present invention seeks in a third aspect to provide a technique for restricting data access in a receiver/decoder.

5

In a third aspect, the present invention provides a method of restricting data access in a receiver/decoder having a memory, the method comprising the steps of:

assigning a plurality of sets of access rights to the data, each set of access rights being assigned to at least one party;

10 storing the data, the sets of access rights and an identifier for each party in a memory of the receiver/decoder;

comparing the identifier of a party requesting access to the data with the or each identifier stored in the memory; and

15 providing the party with the set of access rights assigned thereto in the memory of the receiver/decoder.

Thus, by assigning a plurality of access rights to different parties prior to storing the data in the memory of the receiver/decoder, secure segregation of data in the receiver/decoder can be provided.

20

Preferably, the sets of access rights are stored in a header for the data. This can provide for convenient location of the access rights for the data. The identifiers for the parties may also be stored in the header for the data.

25 The data may be downloaded from a bitstream transmitted by a transmitting system, the sets of access rights and identifiers for the parties being stored within the data at the transmitting system. Thus, this aspect of the present invention extends to a method of restricting access to data broadcast in a digital television system, said method comprising the steps, at a transmitting system, of:

30 assigning a plurality of sets of access rights to the data, each set of access rights being assigned to at least one party;

storing the sets of access rights and identifiers for the parties within the data;

and

transmitting the data;

and, at a receiver/decoder, the steps of:

5 downloading and storing the transmitted data in a memory of the receiver/decoder;

comparing the identifier of a party requesting access to the data with the identifiers stored in the memory; and

providing the party with the set of access rights assigned thereto in the memory of the receiver/decoder.

10

The data may be transmitted in any suitable form, such as in a digital datastream.

15 A further set of access rights may be assigned to at least one party whose identifier is not stored in the memory of the receiver/decoder, a party requesting access to the data being provided with the further set of access rights if the identifier thereof is not stored in the memory. Thus, an unknown or unfavoured party may be prohibited from accessing the data by the selection of a suitable set of access rights.

20 A set of access rights may be assigned to one party only. Preferably, this party is the author of the data. Thus, the author of the data may be assigned a separate set of access rights, which can enable full access to the data by the author.

25 A set of access rights may be assigned to a group of parties, identifiers for each of the members of the group being stored in the memory of the receiver/decoder. Thus, by providing a set of access rights to a group of parties, the number of sets of access rights which need be assigned can be reduced.

30 Preferably a set of access rights is used to determine whether a party is prohibited from reading the data. A set of access rights may also be used to determine whether a party is prohibited from overwriting the data.

The data may be stored in a Flash memory volume of the receiver/decoder.

This aspect of the present invention also extends to apparatus for restricting access to data stored in a memory of a receiver/decoder, a plurality of sets of access rights being assigned to the data, each set of access rights being assigned to at least one party, an identifier for each party being stored in the receiver/decoder, the apparatus comprising:

- 5 means, such as a central processor, for comparing the identifier of a party requesting access to the data with the identifiers stored in the memory; and
- means for providing the party with the set of access rights assigned thereto in the memory of the receiver/decoder. The central processor may also provide the party
- 10 with its assigned set of access rights

This aspect of the invention also provides a receiver/decoder comprising a memory for storing data, a plurality of sets of access rights assigned to the data and an identifier for each party, and apparatus for restricting access to the data as

15 aforementioned.

The receiver/decoder may further comprise a security module for storing the identifiers for the parties.

20 The receiver/decoder may further comprise a receiver for receiving a bitstream including said data, said sets of access rights and said identifiers, and means, such as a demultiplexer and descrambler, for downloading said data, said sets of access rights and said identifiers into said memory. The receiver/decoder is preferably arranged to download MPEG tables.

25

This aspect of the present invention also provides a transmission system comprising:

- means for assigning a plurality of sets of access rights to the data, each set of access rights being assigned to at least one party;
 - means for storing the access rights and identifiers for the parties within the
 - 30 data; and
 - means, such as a transmitter, for transmitting a bitstream including said data.
- The assigning means and the storage means of the transmission system may be

provide by a processing unit of the transmitting-system.

5 The assigning means may be adapted to assign a further set of access rights to parties whose identifiers are not stored in the memory of the receiver/decoder. The assigning means may also be adapted to assign a set of access rights to a group of parties, identifiers for each of the members of the group being stored in said data.

This aspect of the present invention also provides a combination of a receiver/decoder as aforementioned and a transmission system as aforementioned.

10

Various functions of the receiver/decoder may be implemented in hardware, for example in a dedicated integrated circuit; this may provide enhanced speed of operation. Preferably, however, at least some of the functions are implemented in software, preferably implemented by processing means which runs the applications;
15 this can allow greater flexibility, require less components, and allow the receiver/decoder to be updated more readily.

Any of the above features may be combined together in any appropriate combination. Apparatus features may be applied to method aspects and vice versa.

20

Preferred features of the present invention will now be described, by way of example, with reference to the drawings, in which:

25

Figure 1 is a schematic diagram of a known arrangement of memory volumes in a receiver/decoder;

Figure 2 illustrates a prior arborescent organisation of data in a memory volume;

30

Figure 3 illustrates a prior arborescent organisation of data across a plurality of memory volumes;

Figure 4 is a schematic diagram of a digital television system;

Figure 5 is a schematic diagram of the structure of a receiver/decoder of the system of Figure 4;

Figure 6 illustrates the arrangement of a page of Flash memory;

5

Figure 7 illustrates a header of a block of a page of Flash memory;

Figure 8 illustrates the contents of a header block of a file stored in Flash memory;

10 Figure 9 illustrates the arborescence of data stored in Flash memory;

Figure 10 illustrates the contents of the Access mode field in a header block;

15 Figure 11 is a flow diagram illustrating a method of copying blocks of data from one page of Flash memory to another;

Figure 12 illustrates the overall effect of the method illustrated in Figure 11; and

20 Figure 13 is a flow diagram illustrating another method of copying blocks of data from one page of Flash memory to another.

25 An overview of a digital television system 1 is shown in Figure 4. The system includes a mostly conventional digital television system 2 that uses the known MPEG-2 compression system to transmit compressed digital signals. In more detail, MPEG-2 compressor 3 in a broadcast centre receives a digital signal stream (typically a stream of video signals). The compressor 3 is connected to a multiplexer and scrambler 4 by linkage 5.

30 The multiplexer 4 receives a plurality of further input signals, assembles the transport stream and transmits compressed digital signals to a transmitter 6 of the broadcast centre via linkage 7, which can of course take a wide variety of forms including telecommunications links. The transmitter 6 transmits electromagnetic signals via

uplink 8 towards a satellite transponder 9, where they are electronically processed and broadcast via notional downlink 10 to earth receiver 12, conventionally in the form of a dish owned or rented by the end user. The signals received by receiver 12 are transmitted to an integrated receiver/decoder 13 owned or rented by the end user and
5 connected to the end user's television set 14. The receiver/decoder 13 decodes the compressed MPEG-2 signal into a television signal for the television set 14.

Other transport channels for transmission of the data are of course possible, such as terrestrial broadcast, cable transmission, combined satellite/cable links, telephone
10 networks etc.

In a multichannel system, the multiplexer 4 handles audio and video information received from a number of parallel sources and interacts with the transmitter 6 to broadcast the information along a corresponding number of channels. In addition to
15 audiovisual information, messages or applications or any other sort of digital data may be introduced in some or all of these channels interlaced with the transmitted digital audio and video information.

A conditional access system 15 is connected to the multiplexer 4 and the
20 receiver/decoder 13, and is located partly in the broadcast centre and partly in the decoder. It enables the end user to access digital television broadcasts from one or more broadcast suppliers. A smartcard, capable of deciphering messages relating to commercial offers (that is, one or several television programmes sold by the broadcast supplier), can be inserted into the receiver/decoder 13. Using the decoder 13 and
25 smartcard, the end user may purchase commercial offers in either a subscription mode or a pay-per-view mode.

As mentioned above, programmes transmitted by the system are scrambled at the multiplexer 4, the conditions and encryption keys applied to a given transmission
30 being determined by the access control system 15. Transmission of scrambled data in this way is well known in the field of pay TV systems. Typically, scrambled data is transmitted together with a control word for descrambling of the data, the control

word itself being encrypted by a so-called exploitation key and transmitted in encrypted form.

5 The scrambled data and encrypted control word are then received by the decoder 13 having access to an equivalent to the exploitation key stored on a smart card inserted in the decoder to decrypt the encrypted control word and thereafter descramble the transmitted data. A paid-up subscriber will receive, for example, in a broadcast monthly ECM (Entitlement Control Message) the exploitation key necessary to decrypt the encrypted control word so as to permit viewing of the transmission.

10

An interactive system 16, also connected to the multiplexer 4 and the receiver/decoder 13 and again located partly in the broadcast centre and partly in the decoder, enables the end user to interact with various applications via a modem back channel 17. The modem back channel may also be used for communications used in the conditional access system 15. An interactive system may be used, for example, to enable the viewer to communicate immediately with the transmission centre to demand authorisation to watch a particular event, download an application etc.

20 Referring to Figure 5, the elements of the receiver/decoder 13 or set-top box will now be described. The elements shown in this figure will be described in terms of functional blocks.

The decoder 13 comprises a central processor 20 including associated memory elements and adapted to receive input data from a serial interface 21, a parallel interface 22, a modem 23 (connected to the modem back channel 17 of Fig. 4), and switch contacts 24 on the front panel of the decoder.

30 The decoder is additionally adapted to receive inputs from an infra-red remote control 25 via a control unit 26 and also possesses two smartcard readers 27, 28 adapted to read bank or subscription smartcards 29, 30 respectively. The subscription smartcard reader 28 engages with an inserted subscription card 30 and with a conditional access unit 29 to supply the necessary control word to a demultiplexer/descrambler 30 to

enable the encrypted broadcast signal to be desrambled. The decoder also includes a conventional tuner 31 and demodulator 32 to receive and demodulate the satellite transmission before being filtered and demultiplexed by the unit 30.

5 In the case of received audio and video signals, the MPEG packets containing these signals will be demultiplexed and filtered so as to pass real time audio and video data in the form of a packetised elementary stream (PES) of audio and visual data to dedicated audio and video processors or decoders 33, 34. The converted output from the audio processor 33 passes to a preamplifier 35 and thereafter via the audio output
10 of the receiver/decoder. The converted output from the video processor 34 passes via a graphic processor 36 and PAL/SECAM encoder 37 to the video output of the receiver/decoder.

Processing of data within the decoder is generally handled by the central processor 20.
15 The central processor 20 provides a platform having considerable flexibility in enabling an application to communicate with a variety of devices.

As used in this description, an application is a piece of computer code for controlling high level functions of preferably the receiver/decoder 13. For example, when the end
20 user positions the focus of remote control 25 on a button object seen on the screen of the television set 14 and presses a validation key, the instruction sequence associated with the button is run.

An interactive application proposes menus and executes commands at the request of
25 the end user and provides data related to the purpose of the application. Applications may be either resident applications, that is, stored in the ROM (or Flash or other non-volatile memory) of the receiver/decoder 13, or broadcast and downloaded into the RAM or Flash memory of the receiver/decoder 13.

30 Applications are stored in memory locations in the receiver/decoder 13 and represented as resource files. The resource files comprise graphic object description unit files, variables block unit files, instruction sequence files, application files and

data files, as described in more detail in the above-mentioned patent specifications.

With reference to Figure 1, the receiver/decoder contains memory divided into a RAM volume 54, a Flash volume 58 and a ROM volume 56, but this physical organization is distinct from the logical organization. The memory may further be divided into memory volumes associated with the various interfaces. From one point of view, the memory can be regarded as part of the hardware; from another point of view, the memory can be regarded as supporting or containing the whole of the system shown apart from the hardware.

The Flash memory 58 is divided into pages, typically each of many kilobytes in length. Data is stored in Flash memory in the form of blocks of data. Figure 6 illustrates the arrangement of a page of Flash memory. The page 70 comprises a header 72 of length 12 bytes, and blocks 74 of variable length. A block 74 has a size which is determined by the size of the data which is stored in the block.

The header 72 of the page 70 contains, inter alia, the number (NO 76) of the page in the Flash memory and the state (STATE 78) of the page. A page may take one of two stable states:

- (i) an EMPTY state, when the page contains only empty blocks; and
- (ii) a VALID state, when the page contains one or more blocks of data.

The Flash memory 58 contains at least one page having an EMPTY state.

A page may also take one of three temporary states during the copying of the contents of one page to another page (as discussed in more detail below):

- (i) a WRITE state, when contents of a page are being copied to that page from another page;
- (ii) a NEW state, when copying to that page has been completed; and
- (iii) an INVALID state, when the copying of contents from that page to another

page has been completed.

With reference to Figure 7, each block 74 includes a header 80 of length 16 bytes. The header 80 includes the following fields:

5

- (i) an identification (ID 82) of the file to which the block relates;
- (ii) the size (SIZE 84) of the block;
- (iii) the rank identification (RANK 86) of the block; and
- (iv) the status (STATUS 88) of the block.

10

RANK 86 provides a unique identifier of the block when a file is split into a plurality of blocks, each of these blocks being given a respective rank identification. The RANK 86 also serves to specify the position of the block within the file.

15

The block having a RANK=0 comprises a header block for the file. The header block does not include any of the raw data of the file, but instead includes a number of the file attributes. Figure 8 illustrates the contents of a header block 90 of a file. The header block 90 comprises a header 80, in which RANK 86 is set to 0.

20

The header block 90 includes a FatherID field 92. The FatherID field 92 contains the ID 82 of the "father" of the file in the arborescent structure of directories and files stored in the Flash memory 58.

25

Figure 9 illustrates an example of the arborescent structure of blocks stored in the Flash memory 58. The blocks comprise directory blocks and header blocks D, and data blocks F containing the raw data of a file. A directory block is a header block, such as block D2, which is not directly associated with a file but serves to act as a locator for other directory or header blocks.

30

Upon boot-up of the receiver/decoder, the arborescence of the data stored in the Flash memory 58 is read and an outline of the arborescence is stored in the RAM volume 54 in the form of a chart or any other convenient form. The arborescence is

determined by reading the ID 82 and RANK 86 of each data block, and the ID 82, RANK 86 and FatherID 92 of each directory block and header block. For example, header block D1 has ID=1, RANK=0 and FatherID=1 (as it has no "father"). File block F1 has ID=1 and RANK=1. Directory block D2 has ID=2, RANK=0 and FatherID=1 (as its "father" is header block D1 having an ID = 1). Header block D3 has ID=3, RANK=0 and FatherID=2.

In this manner, the arborescence of the data can be quickly generated, as only up to three data fields of each block must be read to generate the arborescence. In addition, each directory block and header block contains an identifier of its father only, and not identifiers of all of its descendants, thus increasing the data storage capacity of a page of Flash memory.

Moreover, when a header block and associated data blocks are moved within the data arborescence, it is only necessary to write the new FatherID in the header of the header block. For example, if header block D11 and associated data blocks F11, F12 and F13 were moved so that the father of header block D11 became header block D6, it would only be necessary to amend the FatherId of header block D11 to FatherId=6. No modification of the headers of previous direct ancestors of header block D11, that is, blocks D1, D8, D9 and D10, or of the new direct ancestors of header block D11, that is, blocks D1, D2, D4, D5 and D6, is required.

When the arborescence of the data stored in the Flash memory changes, the chart stored in the RAM volume 54 is updated when the receiver/decoder is next rebooted.

As discussed earlier, data may be stored in the receiver/decoder either in a single memory volume or across a plurality of memory volumes. The above described arborescent organisation of data in Flash memory, in which header blocks relate to their father, is also applicable to data stored in ROM, RAM or any combination of ROM, RAM and Flash memory.

Referring back to Figure 8, header block 90 includes an Attributes field 92 which

indicates whether the block is a directory block or a header block.

Header block 90 includes Access mode field 94, Owner field 96 and Group field 98. These three fields enable access to the file by various parties to be partially or fully
5 restricted.

The Owner field 96 contains an identification ID of the author of the file. The Group field 98 contains a group identification ID for authorised users of the file. For example, if the file relates to a program for a smartcard reader, the authorised users
10 of the file will include the providers of interactive applications which utilise the program. The identification IDs of each authorised user of the file is stored in a security module in the receiver/decoder.

The contents of the Access mode field 94 determine the extent to which access to the
15 file is permitted by the author of the file, as specified in the Owner field 96, an authorised user of the file, as specified in the Group field 98, and others, such as the manufacturer of the receiver/decoder and a user of the receiver/decoder.

Figure 10 illustrates the Access mode field 94 in more detail. The field 94 comprises
20 six bits, two bits 100, 102 assigned to the author, two bits 104, 106 assigned to the authorised users, and two bits 108 and 110 assigned to others. The values of bits 100, 104 and 108 determine whether reading of the file is prohibited by the author, the authorised users and other respectively, and the values of bits 102, 106 and 110
25 determine whether overwriting of the file is prohibited by the author, the authorised users and other respectively. In the example shown in Figure 10, overwriting of the file by authorised users is prohibited, as is any reading or overwriting of the file by others.

The contents of the Access mode field 94, Owner field 96 and Group field 98 are set
30 prior to the storage of the file in the receiver/decoder. If the file is to be transmitted to the receiver/decoder by the transmitter 6 of the broadcast centre, the contents of the aforementioned fields may be set in the broadcast centre prior to transmission.

Alternatively, the contents of these fields may be set by the author upon writing the file.

5 When access to a file is required, the accessor identifies itself with its identification ID. The central processor 50 determines whether the ID of the accessor corresponds with the ID stored in the Owner field 96. If so, the accessor is identified as the author of the file who is assigned with the set of access rights contained in bits 100 and 102. In the example shown in Figure 10, reading and overwriting of the file by the author is permitted by the central processor 50.

10

If the accessor is not the author of the file, the central processor 50 refers to the security module to determine whether the ID of the accessor is a member of the group of authorised users specified in the Group field 98. If so, the accessor is identified as an authorised user of the file who is assigned with the set of access rights contained in bits 104 and 106. In the example shown in Figure 10, the authorised user is permitted only to read the file by the central processor 50.

15

If the accessor is not an authorised user of the file, then the accessor is identified as an "other" who is assigned with the set of access rights contained in bits 108 and 110.

20

In the example shown in Figure 10, any access to the file by the accessor is prohibited by the central processor.

25

The values of the bits 100 to 110 can only be changed by overwriting of the file. In the example shown in Figure 10, only the author of the file is permitted to overwrite the file. Thus, the provision of the Access mode field 94, the Owner field 96 and the Group field 98 in the header block 90 of the file provides for a simple but yet effective security system for the file.

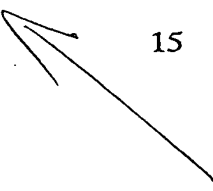
30

With reference to Figure 9, the header block 90 also includes a Version field 112 which contains the version number of the file, and a Name field 114 which contains the name of the file in ASCII code.

Referring back to Figure 7, the STATUS field 88 of a block of a page of the Flash memory may take one of three stable states:

- (i) a FREE state, when the block is empty; and
 - 5 (ii) an INVALID state, when a new version of the block has been copied to another page of the Flash memory, or when the file has been deleted; and
 - (iii) a VALID state, when a previous version of the block has been set to an INVALID state.
- 10 Only usually the last block of the page is in the FREE state, and represents the amount of memory space available for storing fresh data in that page.

A block may also take one of three temporary states during the writing of the contents of the block, or the copying of the contents of the block, via RAM, to another page (as discussed in more detail below):

- 
- (i) a CREATE state, when the header of the block is being written;
 - (ii) a WRITE state, when the remainder of the contents of the block is being written; and
 - 20 (iii) a NEW state, when the block has been completed but writing or copying of additional blocks of the file has yet to be completed.

As mentioned earlier, one of the pages of Flash memory is maintained in an EMPTY state. In time, all of the remaining pages of Flash memory will become full with

25 VALID and INVALID blocks of data, and so it will become necessary to effectively delete one or more of the blocks of INVALID data to provide enough space for a block containing fresh data. However, party blocks of data cannot be overwritten with new data; an entire page of Flash memory must be erased to enable that page to be re-used.

30

In order to provide enough space for a new file to be written in Flash memory, in overview the VALID blocks only of a VALID page are copied, via RAM, to an

EMPTY page, the INVALID blocks of the VALID page not being copied to the EMPTY page. The VALID blocks are copied to the top of the EMPTY page in order to leave enough room at the foot of the page to store the fresh data. The contents of the VALID page are erased, the EMPTY page becomes a VALID page and the previous VALID page becomes the EMPTY page.

A method of "cleaning" the contents of the Flash memory when fresh data is to be stored in the Flash memory is described in more detail below, with reference to Figures 11 and 12.

10

In step 100, the central processor 50 reviews the headers 80 of each of the blocks 74 of each of the VALID pages of the Flash memory in order to identify the VALID page with the highest number of FREE or INVALID blocks.

15

In step 102, the VALID page having the highest number of FREE or INVALID blocks is selected.

In step 104, the status of the empty page is changed from EMPTY to WRITE, and the number of the selected page is stored temporarily in the header of the WRITE page.

20

In step 106, the central processor 106 determines whether there are any VALID blocks in the VALID page. If there are one or more VALID blocks in the VALID page, in step 108, the first VALID block is copied, via RAM, into a FREE block of the WRITE page. In more detail, the state of the FREE block is changed to CREATE whilst the header of the VALID block is being copied, then to WRITE whilst the contents of the VALID block is being copied, and then to NEW when the copying of the contents of the block has been completed. When the copying of the contents of the block has been completed, in step 110 the state of the copied VALID block in the VALID page is changed to INVALID.

30

In step 112, the central processor 50 determines whether there are any more VALID blocks in the VALID page. If there is a second VALID block, in step 114 the central

processor determines whether the second VALID block has the same ID 82 as the NEW block in the WRITE page. If it does, in step 116 the state of the NEW block in the WRITE page is maintained and the second VALID block is copied, via RAM, into a FREE block in the WRITE page directly beneath the NEW block. If it does not have the same ID 82 as the NEW block in the EMPTY page, then in step 118 the status of the NEW block is changed to VALID and the second VALID block is copied, via RAM, into a FREE block in the WRITE page directly beneath the VALID block in the WRITE page. In step 120, the state of the second VALID block in the VALID page is changed to INVALID.

Steps 112 to 120 are repeated as necessary until the central processor 50 determines that there are no more VALID blocks in the VALID page. Then, in step 122, the statuses any remaining NEW blocks in the VALID page are changed to VALID, so that the WRITE page contains only VALID and FREE blocks, and the VALID page contains only INVALID and, if any, FREE blocks.

In step 124, the status of the WRITE page is changed to NEW. In step 126, the status of the VALID page is changed to INVALID. In step 128, the status of the NEW page is changed to VALID, and finally in step 130, the INVALID page is erased and the status of the erased page changed to EMPTY, so that there is once again at least one EMPTY page of Flash memory.

At the end, all blocks of the new Page have the same state that the blocks of the Erased Page had. (NEW or VALID blocks only)

The overall effect of the above method is summarised in Figure 12. The VALID blocks V of the VALID page are effectively moved to the top of the EMPTY page in order to provide storage space in the EMPTY page for fresh data, once the status of the EMPTY page has been changed to VALID.

A potential problem with the above method is that, due to the variable nature of the size of each block, a large number of relatively small INVALID blocks may have been effectively erased, with the result that the available space at the bottom of the new VALID page is insufficient to store completely the fresh data. Figure 13 illustrates an alternative method of "cleaning" the contents of the Flash memory.

FALSE : in Figure 12 we select the page with the most Invalid and free blocks. The "most" means that no other page has more space. So looking for "appropriate cumulative size can't give more space than in Figure 12.

In step 200, the storage space required to store completely the fresh data in a page of Flash memory is determined by the central processor 50. Then, in step 202, the central processor identifies a VALID page in which the cumulative size of the INVALID blocks is sufficient to store the fresh data. The method then proceeds with steps 204 to 230, as required, which are identical to steps 104 to 130.

With this alternative method, the "cleaning" operation of the Flash memory 58 results in sufficient space to store the fresh data in a page of Flash memory 58.

It will be understood that the present invention has been described above purely by way of example, and modifications of detail can be made within the scope of the invention. Each feature disclosed in the description, and (where appropriate) the claims and drawings may be provided independently or in any appropriate combination.

CLAIMS

Restricting Access to Data

- 5 1. A method of restricting data access in a receiver/decoder having a memory, the method comprising the steps of:
 - assigning a plurality of sets of access rights to the data, each set of access rights being assigned to at least one party;
 - 10 storing the data, the sets of access rights and an identifier for each party in a memory of the receiver/decoder;
 - comparing the identifier of a party requesting access to the data with the or each identifier stored in the memory; and
 - providing the party with the set of access rights assigned thereto in the memory of the receiver/decoder.
- 15 2. A method according to Claim 1, wherein the sets of access rights are stored in a header for the data.
3. A method according to Claim 2, wherein the identifiers for the parties are
20 stored in the header for the data.
4. A method according to Claim 2 or 3, wherein the data is downloaded from a bitstream transmitted by a transmitting system, the sets of access rights and identifiers for the parties being stored within the data at the transmitting system.
- 25 5. A method of restricting access to data broadcast in a digital television system, said method comprising the steps, at a transmitting system, of:
 - assigning a plurality of sets of access rights to the data, each set of access rights being assigned to at least one party;
 - 30 storing the sets of access rights and identifiers for the parties within the data;
 - and
 - transmitting the data;

and, at a receiver/decoder, the steps of: -

downloading and storing the transmitted data in a memory of the receiver/decoder;

5 comparing the identifier of a party requesting access to the data with the identifiers stored in the memory; and

providing the party with the set of access rights assigned thereto in the memory of the receiver/decoder.

6. A method according to Claim 5, wherein the data is transmitted in a digital
10 datastream.

7. A method according to any preceding claim, in which a further set of access rights is assigned to at least one party whose identifier is not stored in the memory of the receiver/decoder, a party requesting access to the data being provided with the
15 further set of access rights if the identifier thereof is not stored in the memory.

8. A method according to any preceding claim, wherein a set of access rights is assigned to one party only.

20 9. A method according to Claim 8, wherein said one party is the author of the data.

10. A method according to any preceding claim, wherein a set of access rights is assigned to a group of parties, identifiers for each of the members of the group being
25 stored in the memory of the receiver/decoder.

11. A method according to any preceding claim, wherein a set of access rights is used to determine whether a party is prohibited from reading the data.

30 12. A method according to any preceding claim, wherein a set of access rights is used to determine whether a party is prohibited from overwriting the data.

13. A method according to any preceding claim, wherein the data is stored in a Flash memory volume of the receiver/decoder.

5 14. Apparatus for restricting access to data stored in a memory of a receiver/decoder, a plurality of sets of access rights being assigned to the data, each set of access rights being assigned to at least one party, an identifier for each party being stored in the receiver/decoder, the apparatus comprising:

means for comparing the identifier of a party requesting access to the data with the identifiers stored in the memory; and

10 means for providing the party with the set of access rights assigned thereto in the memory of the receiver/decoder.

15 15. A receiver/decoder comprising a memory for storing data, a plurality of sets of access rights assigned to the data and an identifier for each party, and apparatus for restricting access to the data as claimed in Claim 14.

16. A receiver/decoder according to Claim 15, wherein the data is stored in a Flash memory volume of the receiver/decoder.

20 17. A receiver/decoder according to Claim 15 or 16, wherein the access rights are stored in a header for the data.

25 18. A receiver/decoder according to Claim 17, wherein the identifiers for the parties are stored in the header for the data.

19. A receiver/decoder according to Claim 15 or 16, further comprising a security module for storing the identifiers for the parties.

30 20. A receiver/decoder according to any of Claims 15 to 19, further comprising a receiver for receiving a bitstream including said data, said sets of access rights and said identifiers, and means for downloading said data, said sets of access rights and said identifiers into said memory.

21. A receiver/decoder according to Claim 20, arranged to download MPEG tables.
22. A transmission system comprising:
means for assigning a plurality of sets of access rights to the data, each set of
5 access rights being assigned to at least one party;
means for storing the access rights and identifiers for the parties within the
data; and
means for transmitting a bitstream including said data.
- 10 23. A transmission system according to Claim 22, wherein the assigning means is
adapted to assign a further set of access rights to parties whose identifiers are not
stored in the memory of the receiver/decoder
24. A transmission system according to Claim 22 or 23, wherein the assigning
15 means is adapted to assign a set of access rights to a group of parties, identifiers for
each of the members of the group being stored in said data.
25. A transmission system according to any of Claims 22 to 24, wherein the access
rights are used to determine whether a party is prohibited from reading the data.
- 20 26. A transmission system according to any of Claims 22 to 24, wherein the access
rights are used to determine whether a party is prohibited from overwriting the data.
27. A transmission system according to any of Claims 22 to 26, wherein the
25 transmitting means is adapted to transmit the data in MPEG format.
28. A combination of a receiver/decoder according to any of Claims 15 to 21 and
a transmission system according to any of Claims 22 to 27.

Recording arborescence of data

1. A method of recording the arborescence of data stored as files and directories,
5 said method comprising the step of storing in association with each file and directory of the data an identifier of the directory, if any, immediately preceding that file or directory in the arborescence of the data.
2. A method according to Claim 1, wherein the identifier is stored in a header of
10 the file or directory.
3. A method according to Claim 2, wherein at least part of the data is stored in a Flash memory volume.
- 15 4. A method according to Claim 3, wherein the arborescence of that part of the data is preferably recorded in a RAM memory volume.
5. A method according to Claim 3 or 4, wherein the header of a file is stored in a dedicated block of Flash memory.
20
6. A method according to any preceding claim, wherein the stored identifier is changed when a file or directory is moved to immediately precede another directory.
7. A method according to any preceding claim, wherein the data is stored in a
25 receiver/decoder.
8. Apparatus for recording the arborescence of data stored as files and directories, said apparatus comprising means for storing in association with each file and directory of the data an identifier of the directory, if any, immediately preceding that file or
30 directory in the arborescence of the data.
9. Apparatus according to Claim 8, wherein the storage means is adapted to store

the identifier in a header of the file or directory.

10. Apparatus according to Claim 8 or 9, in combination with a Flash memory volume and a RAM memory volume.

5

11. Apparatus according to Claim 10, further comprising means for recording in the RAM memory volume the arborescence of data stored in the Flash memory volume.

10 12. Apparatus according to any of Claims 8 to 11, further comprising means for

13. A receiver/decoder comprising apparatus according to any of Claims 8 to 12.

Organising blocks of Flash memory

1. A method of transferring blocks of data between pages of memory to enable
5 fresh data to be stored in said memory, said memory comprising a plurality of pages,
each page of said memory comprising a plurality of blocks, said blocks comprising
free blocks which contain no data, valid blocks containing valid data and invalid
blocks containing invalid data, at least one of the pages of memory being designated
as a transfer page initially comprising only free blocks, the method comprising the
10 steps of:
 copying valid blocks from a designated page of memory into free blocks at one
end of said transfer page; and
 erasing the designated page.
- 15 2. A method according to Claim 1, wherein the data is subsequently stored in at
least one free block after the copied valid pages in the transfer page.
3. A method of storing data in memory, said memory comprising a plurality of
pages, each page comprising a plurality of blocks, said blocks comprising free blocks
20 which contain no data, valid blocks containing valid data and invalid blocks containing
invalid data, at least one of the pages of memory being designated as a transfer page
initially comprising only free blocks, the method comprising the steps of:
 copying valid blocks from a designated page of memory into free blocks at one
end of said transfer page;
25 erasing the designated page; and
 storing the data in at least one free block after the copied valid blocks in said
transfer page.
4. A method according to any preceding claim, wherein the erased designated
30 page is redesignated as a new transfer page.
5. A method according to any preceding claim, wherein a page is designated in

dependence on the number of invalid blocks of the page.

6. A method according to Claim 5, wherein the designated page is the page having the most invalid blocks.

5

7. A method according to any of Claims 1 to 4, wherein a page is designated in dependence on the cumulative size of the invalid blocks of the page.

8. A method according to Claim 7, wherein the designated page is a page having at least one invalid block, said at least one invalid block having a cumulative size equal to or greater than the size of the fresh data.

10

9. A method according to any preceding claim, wherein the blocks have a variable size.

15

10. A method according to any preceding claim, wherein the memory comprises a Flash memory volume.

11. A method according to any preceding claim, wherein the memory comprises a memory volume of a receiver/decoder.

20

12. A method according to Claim 11, wherein the data is downloaded from a bitstream.

13. A method according to Claim 12, wherein the data is formatted in the form of MPEG tables.

25

ABSTRACT

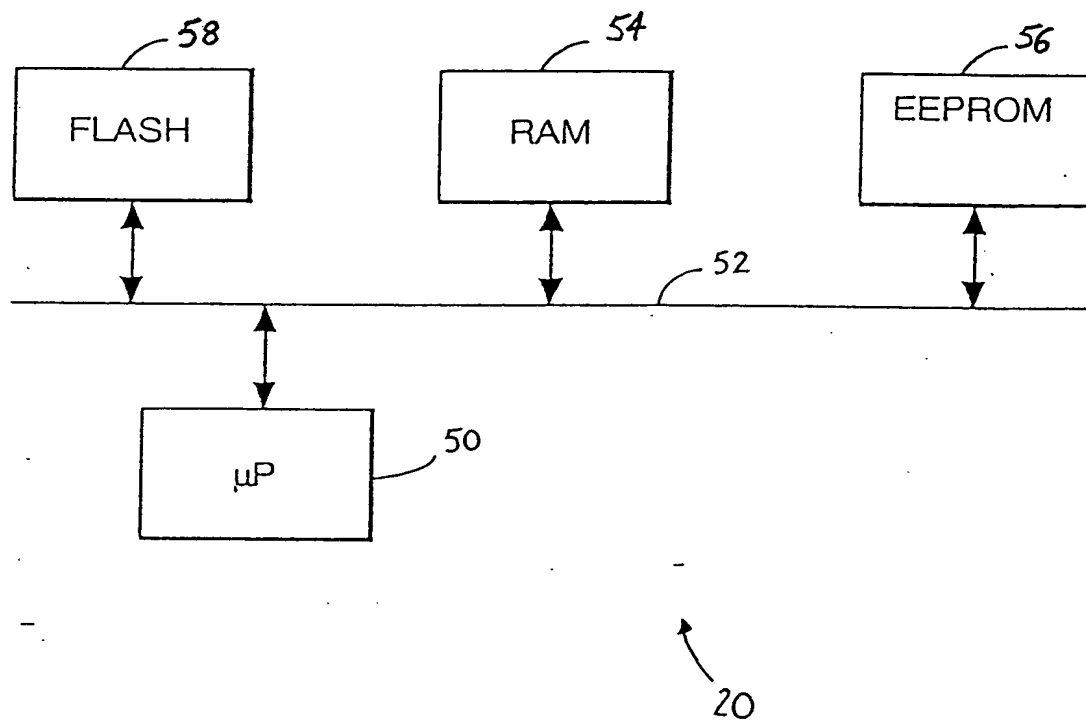
MANAGEMENT OF DATA IN A RECEIVER/DECODER

- 5 In order to restrict data access in a receiver/decoder, a plurality of sets of access rights are assigned to the data, each set of access rights being assigned to at least one party. The data, the sets of access rights and an identifier for each party are stored in a memory of the receiver/decoder. The identifier of a party requesting access to the data is compared with the or each identifier stored in the memory, and the party
- 10 provided with the set of access rights assigned thereto in the memory of the receiver/decoder.

(no figure)

PRIOR ART

Fig. 1.



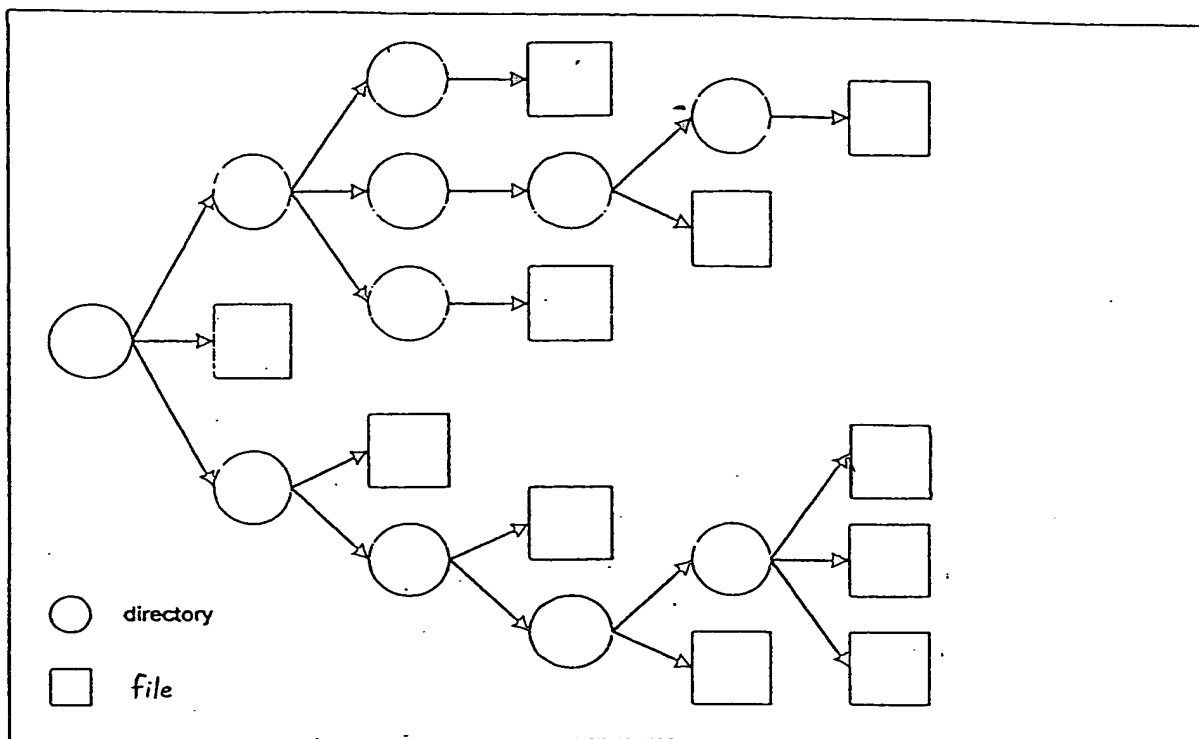


Fig. 2 PRIOR ART

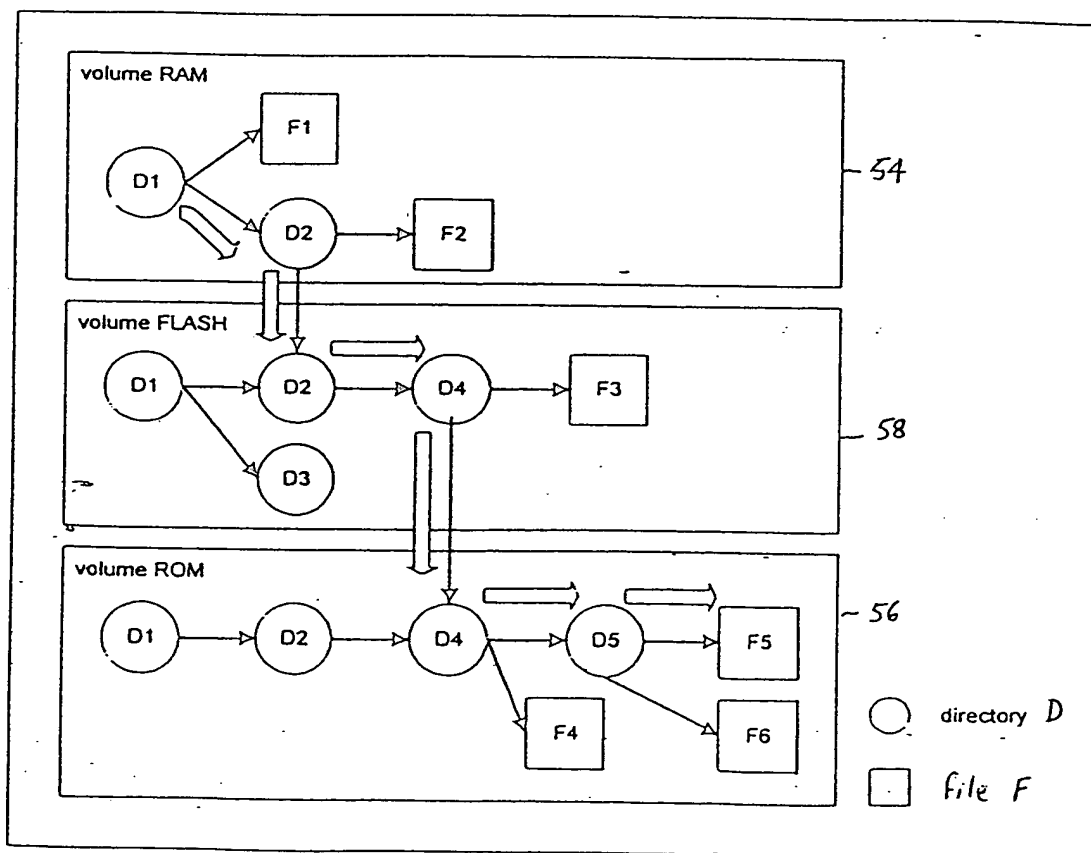


Fig 3 PRIOR ART

Fig. 4.

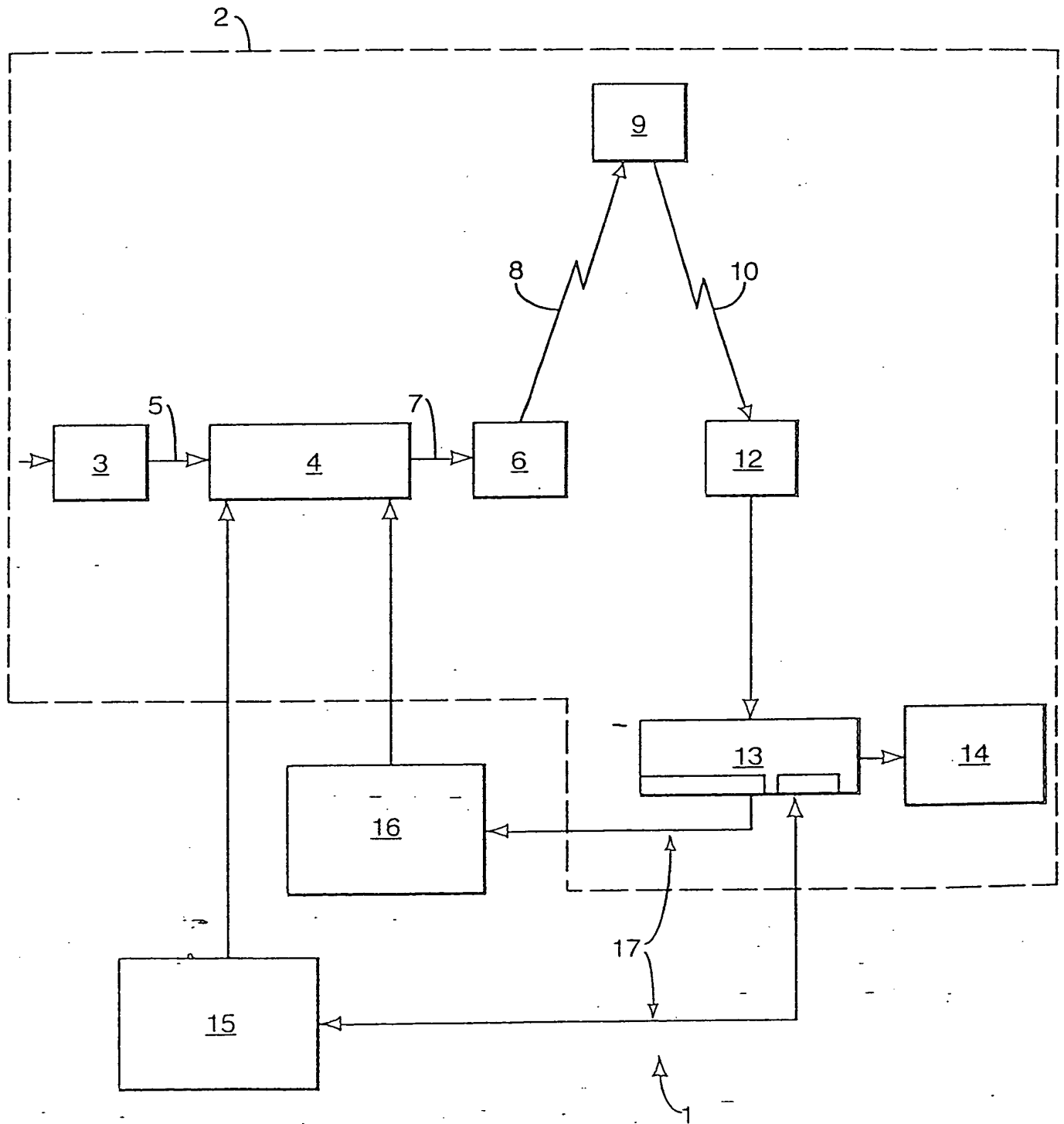
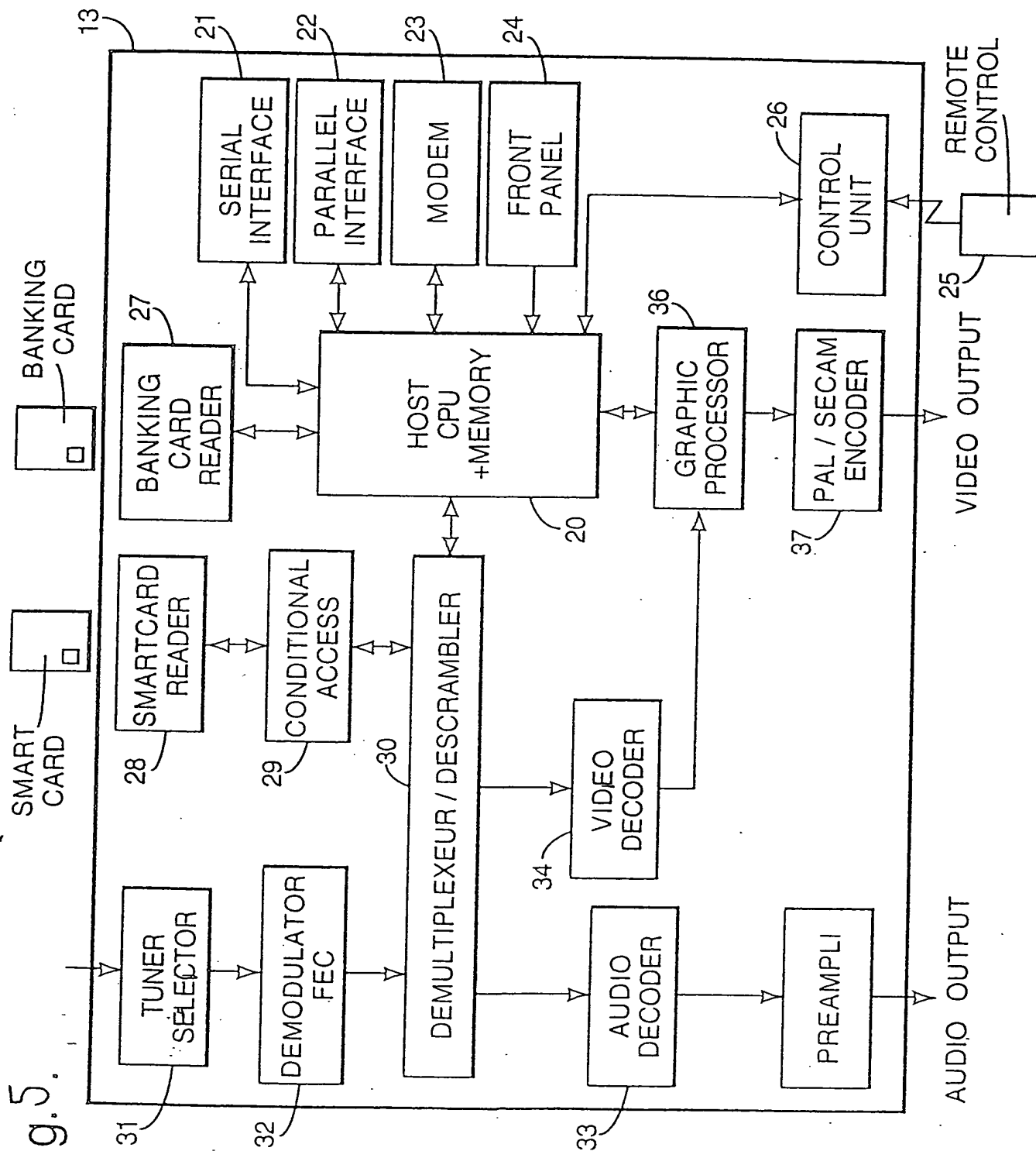


Fig. 5.



No	STATE	

FIG. 6.

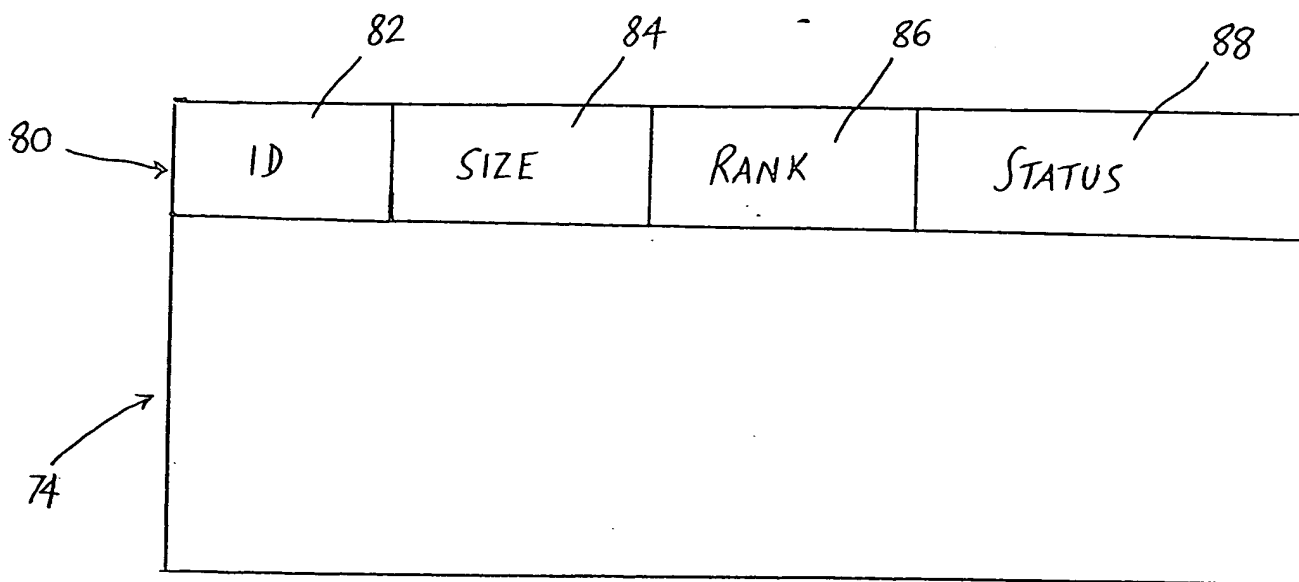


FIG. 7.

90

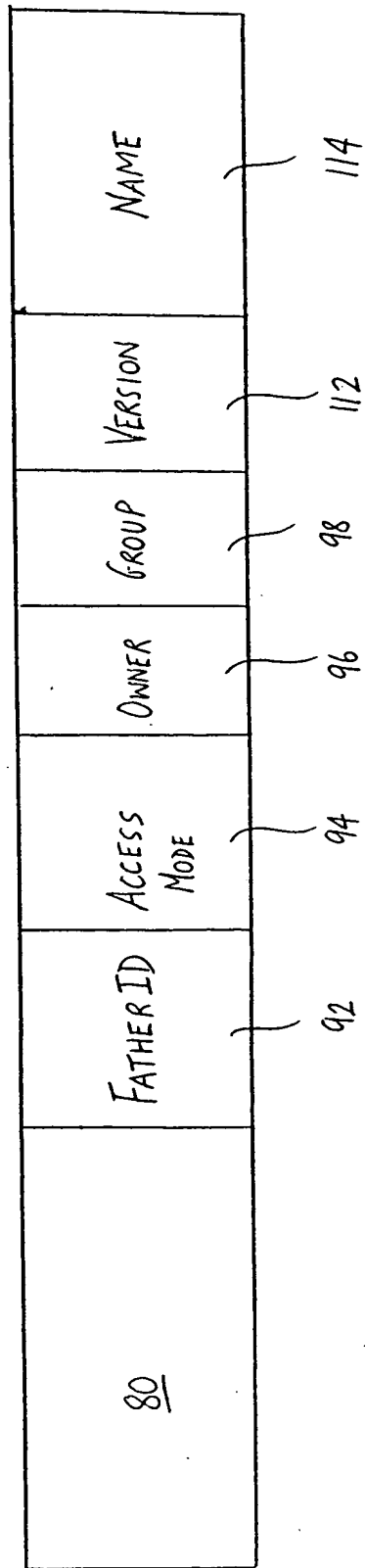


FIG. 8

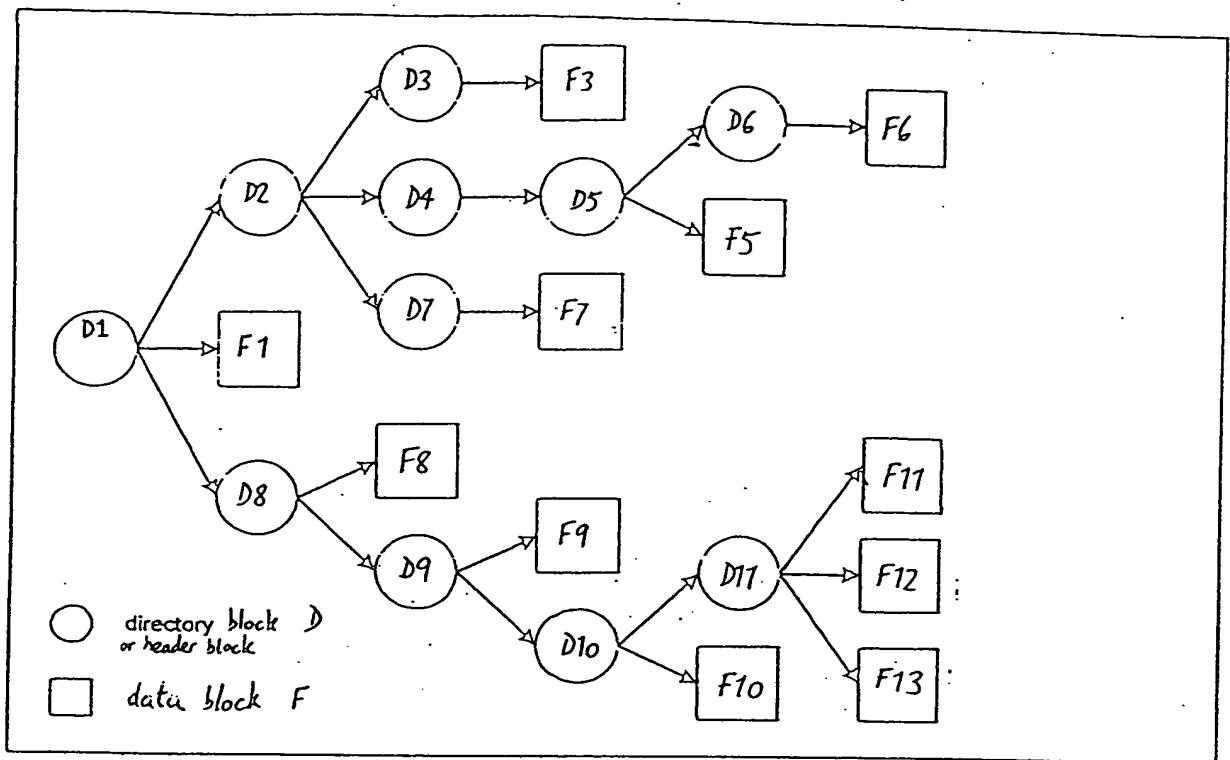


Fig. 9

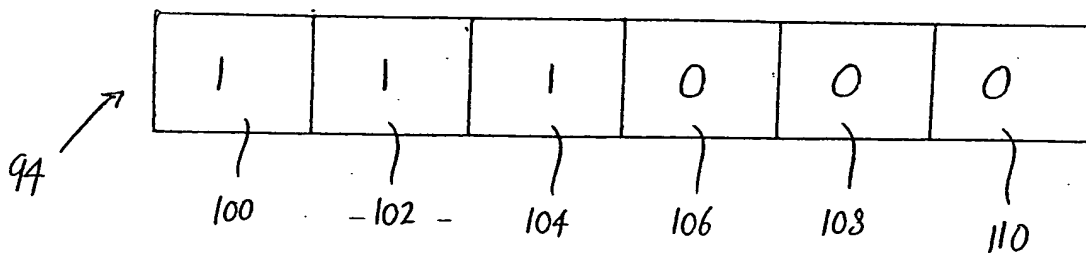


Fig. 10

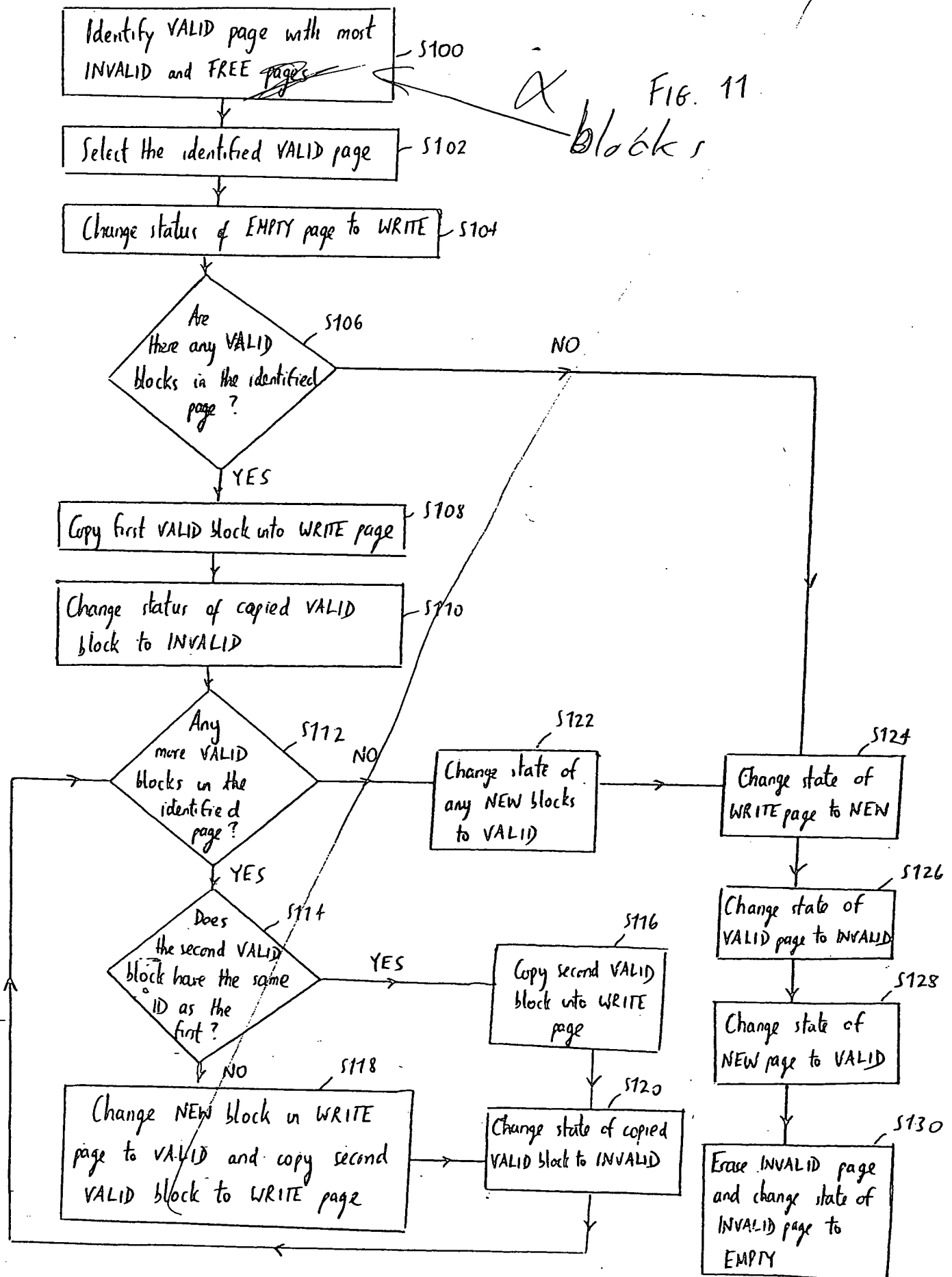


FIG 13

